

Also possible to use RSA to build PKE:

"Textbook RSA" (How NOT to encrypt): Consider the following candidate of a PKE scheme from RSA:

- Setup: Sample (N, e, d) where $N = pq$ and $ed = 1 \pmod{\varphi(N)}$. Output $pk = (N, e)$ and $sk = (N, d)$
 - Encrypt (pk, m) : Output $c \leftarrow m^e$
 - Decrypt (sk, ct) : Output $m \leftarrow c^d$
- } Correct since $c^d = (m^e)^d = m^{ed} = m^1 = m \pmod{N}$

Correctness follows from correctness of TDP.

How about security? NO. 1. RSA says that computing e^{th} root of random element should be difficult

↳ Does not apply if messages chosen adversarially (e.g., semantic security definition)

↳ Does not say anything about hiding preimage (e.g., x^e can leak information about x so long as leakage is not sufficient to fully recover x - this is a weaker property than full indistinguishability)

2. This scheme is deterministic: cannot be semantically secure!

NEVER use textbook RSA!

↳ in fact, vulnerable to message-recovery attacks in many settings

To use RSA to construct a PKE scheme, we will use a similar strategy as in the FDH signature construction:

- Setup: Sample $N = pq, e, d$ where $ed = 1 \pmod{\varphi(N)}$. $pk = (N, e)$, $sk = d$

- Encrypt (pk, m) : Sample $x \xrightarrow{R} \mathbb{Z}_N^*$

Scheme is randomized!

Let $k \leftarrow H(x)$ where $H: \mathbb{Z}_N^* \rightarrow K$ is an (ideal) hash function and K is the key-space for an symmetric authenticated encryption scheme

Compute $y \leftarrow x^e$ and $ct' \leftarrow \text{Enc}_{AE}(k, m)$

Output (y, ct')

- Decrypt $(sk, ct' = (y, ct'))$: Compute $x = y^d \pmod{N}$, $k \leftarrow H(x)$, and output $m \leftarrow \text{Dec}_{AE}(k, ct')$

This is an example of hybrid encryption or KEM: y is used to encapsulate the key and ct' is an encryption under k

Theorem. If the RSA assumption holds and H is modeled as a random oracle, then the above encryption scheme is semantically secure. [In fact, this scheme is CCA-secure in the random oracle model]

Proof intuition. Given a ciphertext (y, ct') and public key $pk = pp$:

- Adversary cannot compute x from y (by RSA - observe that x is uniform over \mathbb{Z}_N^*)
- Adversary cannot evaluate H on x , so k is uniformly random and hidden from adversary
- Semantic security follows from semantic security of symmetric encryption scheme.

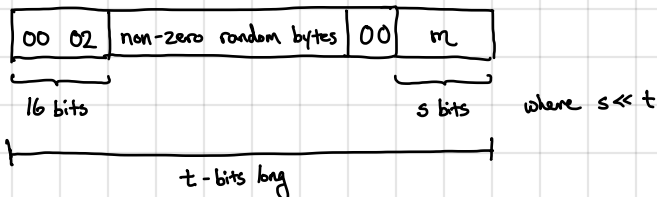
In practice: Most widely-used standard for RSA encryption is PKCS1 (by RSA labs)

↳ Has shorter ciphertexts if we are encrypting a single \mathbb{Z}_N element (no need for KEM + symmetric component)
(helpful if PKE just used to encrypt short token or metadata)

General approach: suppose N is 2048 bits and we want to encrypt 256-bit messages

↳ we will first apply a randomized padding to m to obtain a 2048-bit padded message

PKCS1 padding:
(mode 2)



Encryption: Compute $m_{\text{pad}} \leftarrow \text{PKCS}(m)$ and set $c \leftarrow m_{\text{pad}}^e$ [i.e., directly apply RSA trapdoor permutation to padded message]

Decryption: Compute $m_{\text{pad}} \leftarrow c^d$ and recover m from m_{pad}

In SSL v3.0: during the handshake, server decrypts client's message and checks if resulting m_{pad} is well-formed (i.e., has valid PKCS1 padding) and rejects if not

↳ scheme is vulnerable to a chosen-ciphertext attack!

↳ allows adversary to eavesdrop on connection

Devastating attack on SSL 3.0 and very hard to fix: need to change both servers + clients!

↳ TLS 1.0: fix is to set $m \leftarrow \mathbb{Z}_N^*$ if decryption ever fails and proceed normally (never alert client if padding is malformed) — setup fails at a later point in time, but hopefully no critical information is leaked...

Take-away: PKCS1 is not CCA-secure which is very problematic for key exchange

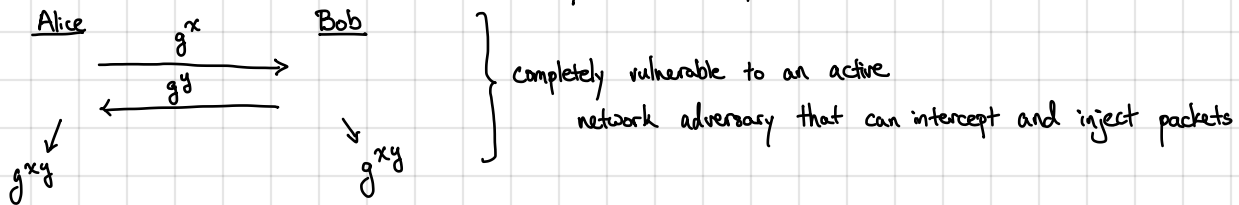
↳ Absence of security proof should always be troubling...

New standard: Optimal Asymmetric Encryption Padding (OAEP) [1994] } standardized in PKCS1

↳ Can be shown to be CCA-secure in random oracle model

version 2.0

Now that we have digital signatures, let's revisit the question of key exchange (with active security)

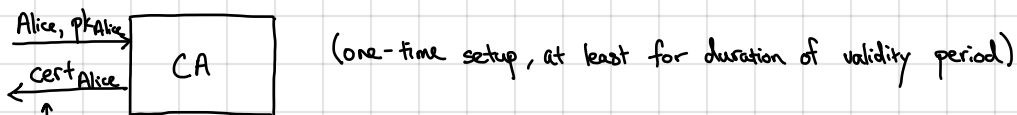


In addition, should guarantee that one compromised session should not affect other honest sessions

- Alice \leftrightarrow Eve should not compromise security of Alice \leftrightarrow Bob

Authenticated key exchange (AKE): provides security against active adversaries

- Requires a "root of trust" (certificate authority) \rightarrow we need some binding between keys and identities



\leftarrow the certificate binds Alice's public key pk_{Alice} to Alice's identity

- Certificates typically have the following format (X509):

- Subject (entity being authenticated)
- Public key (public key for subject for signature scheme)
- CA: identity of the CA issuing the certificate
- Validity dates for certificate
- CA's signature on certificate

\leftarrow the browser and operating system have a set of hard-coded certificate authorities and their respective public keys (usually several hundred authorities)

[public-key infrastructure (PKI)]